Graph Theory

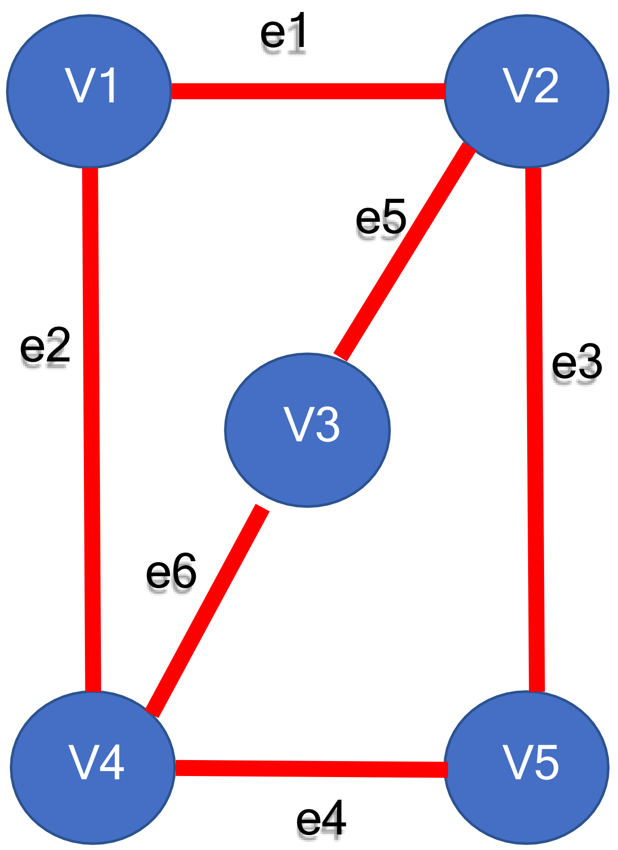
- Course 1 -

Chapter 1: VOCABULARY(1/1)

Finite graph

- Graph (*graph*)
  - Finite graph (*finite graph*)

A finite graph G = (V,E) is defined by the finite set
V ={v1, v2, . . . , vn} whose elements are called ***Vertices***,
and by the finite set
E={e1, e2, . . . , em} whose elements are called **Edges**
(

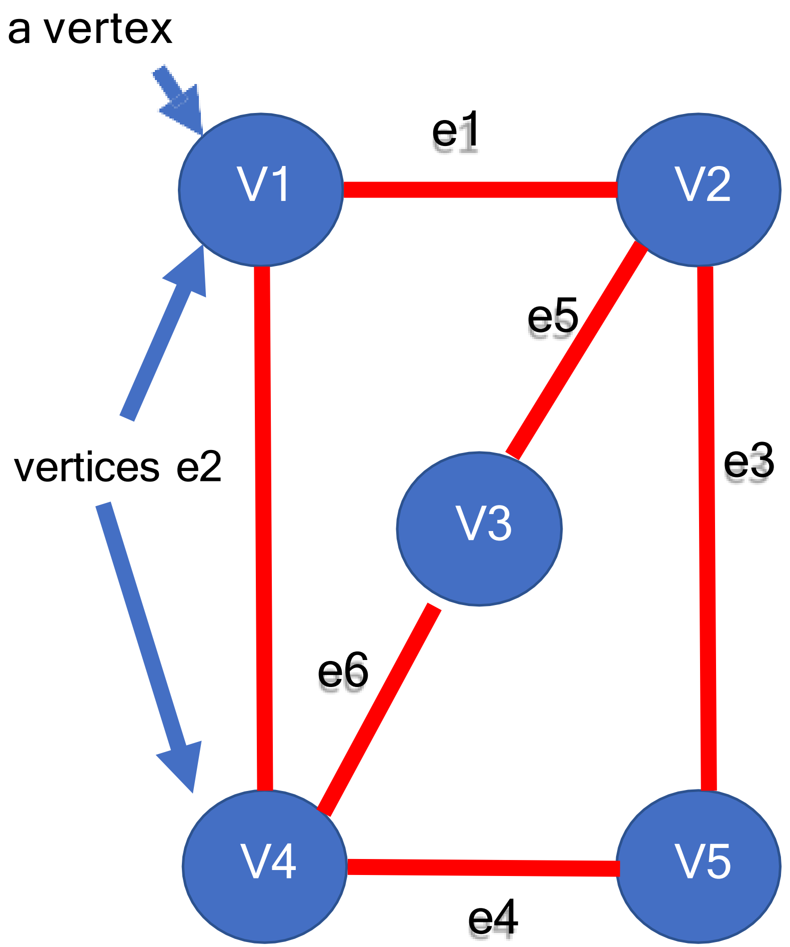


Finite graph

- Graph (*graph*)
  - Finite graph (*finite graph*)

The finite graph G = (V,E) is defined by the finite set of ***Vertices*** V =
{v1, v2, V3, V4, V5} and the finite set of ***Edges***

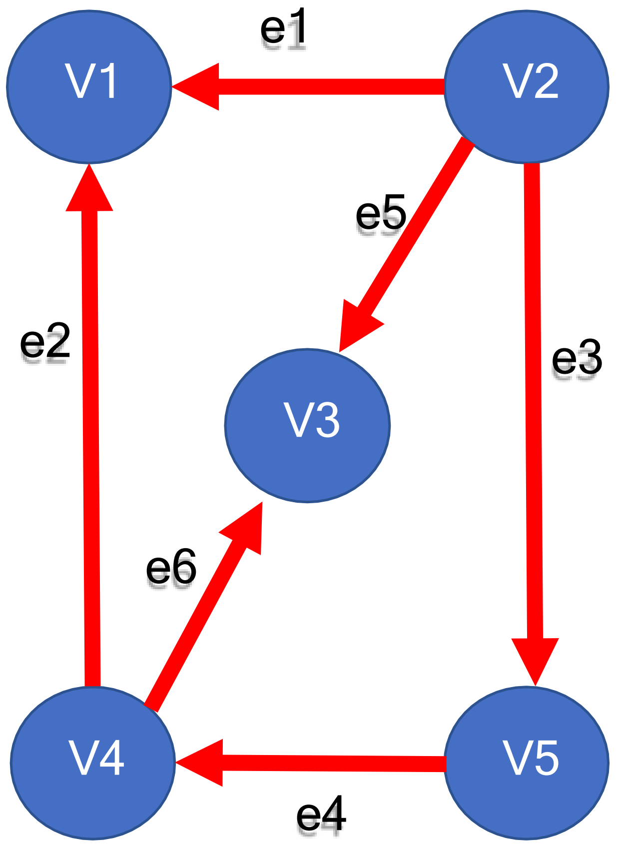
E={e1, e2,e3,e4,e5,e6}



Directed graph

- Directed graph (*directed graph*)

A finite graph G = (V,E) is defined by the finite set
 V ={v1, v2, . . . , vn} whose elements are called ***Vertices***,
and by the finite set
E={e1, e2, . . . , em} whose elements are called **Arcs**
(**directed or oriented *Edges***)....

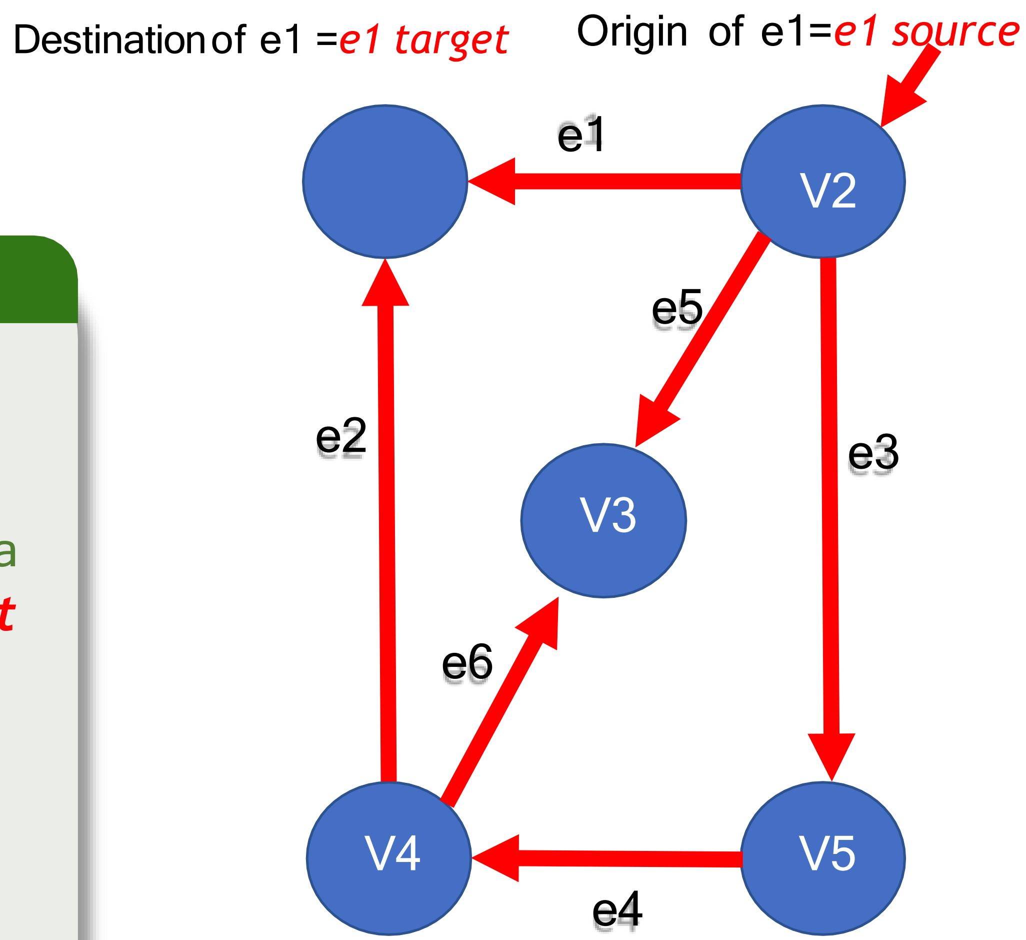


Directed graph

- Directed graph, digraph (*directed graph*)

Each **Directed Edge** has a **origin(*source*)** and one ***target* *(destination)***

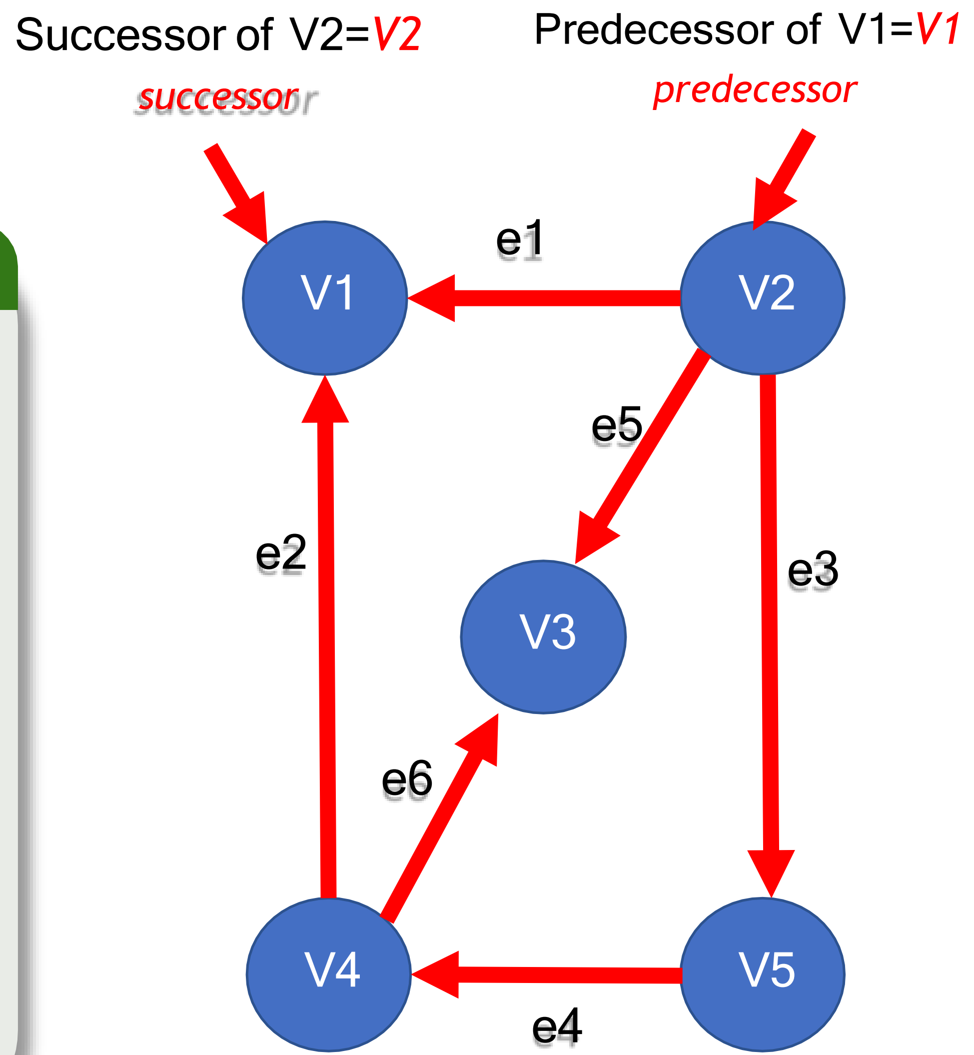
The directed graph is also called a <u>digraph</u>.



Directed graph

- Successor and Predecessor in a Directed Graph

Each **Vertex** has a set of ***predecessors*** and a set of **successors**



Directed/undirected graph

- Cardinality of vertices and arcs/ edges (*vertices cardinality or graph order*)

In a finite graph G = (V,E) defined by the finite set of ***Vertices*** V

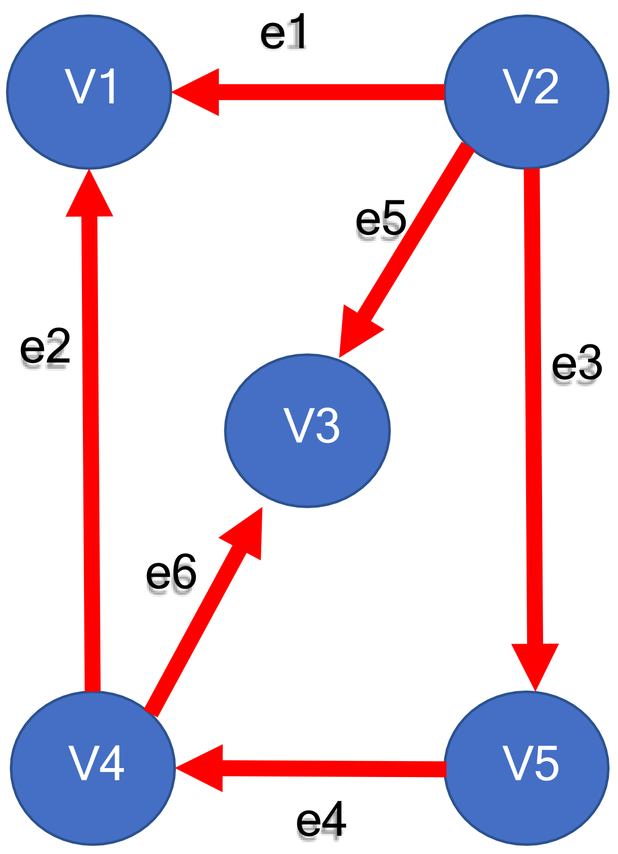
= {v1, v2, V3, V4, V5} and the finite set of **edges**

E={e1, e2,e3,e4,e5,e6}
|V| = n (n is the number of vertices of G) called **order of G *(G Order)***

|E| = m (m is the number of arcs of G).

Order of G= 5

|E| = 6



Undirected graph

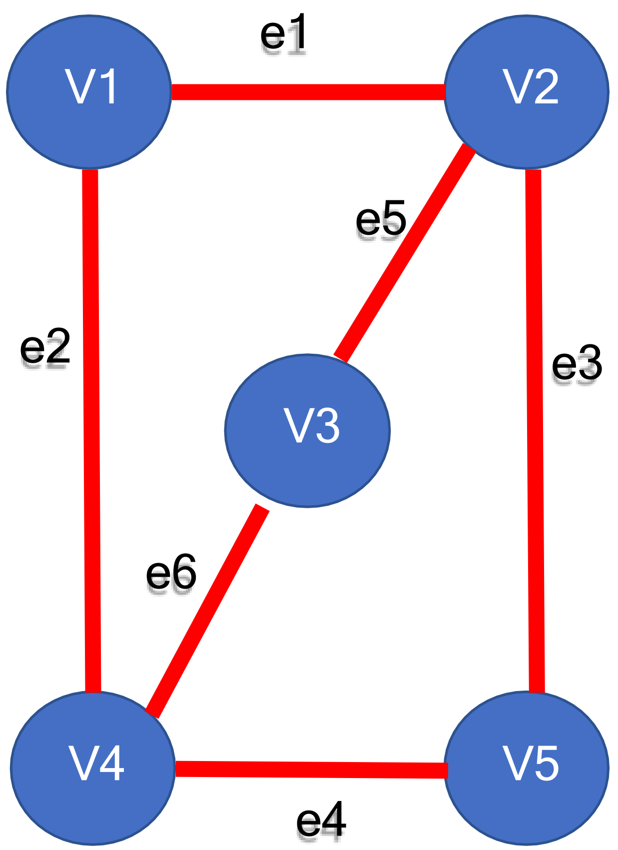
- Degree of a vertex (*vertex degree*)

In an undirected graph G = (V,E) defined by the finite set of ***Vertices*** V ={v1, v2,v3,v4,v5} and the finite set of **edges**

E={e1, e2,e3,e4,e5,e6}
The **degree** of vertex (***vertex degree***) is the number of edges incident to it.

Degree of V1 = 2

Degree of V2 = 3



Directed graph

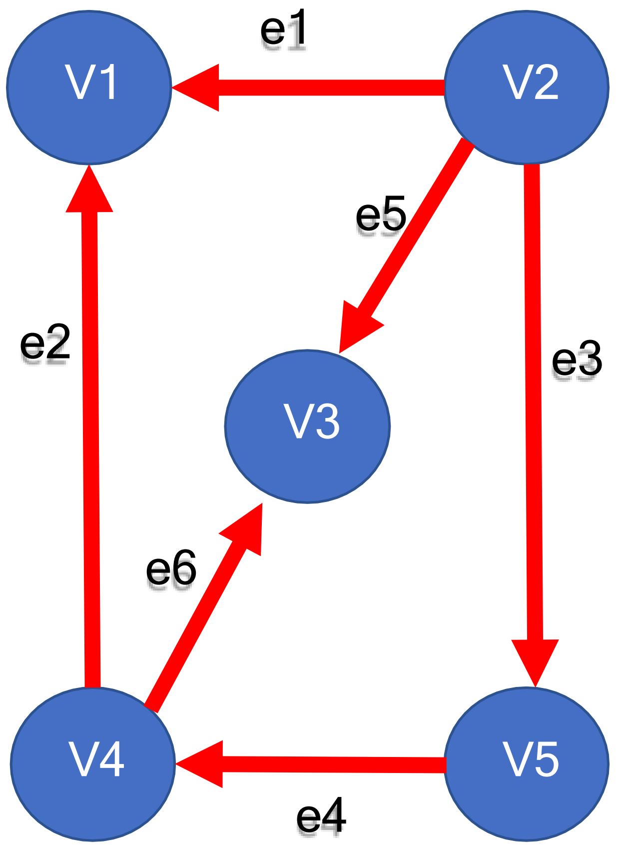
- Degree of a vertex (*vertex degree*)

In a directed graph G = (V,E) defined by the finite set of ***Vertices*** V ={v1, v2,v3,v4,v5} and the finite set of ***Edges*** E={e1, e2,e3,e4,e5,e6}

The **degree** of a ***vertex*** (***vertex degree***) is the sum of the number of incoming arcs and the number of outgoing arcs.

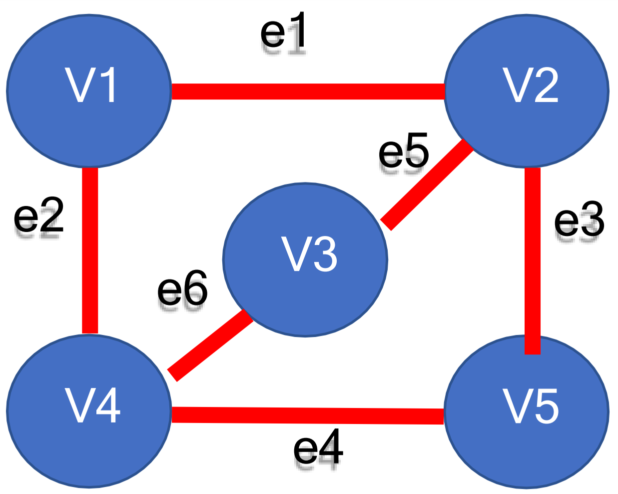
Degree of V5 = 1+1=2

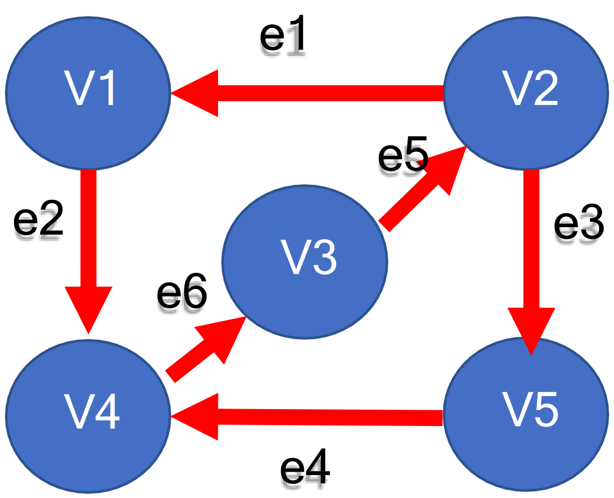
Degree of V2 = 0+3=3



Directed/undirected graph

- Chain vs Path
  - chain (*chain*)==undirected graph
    - Elementary chain vs simple chain
  - Path (*path*)==directed graph
    - Elementary path vs simple path





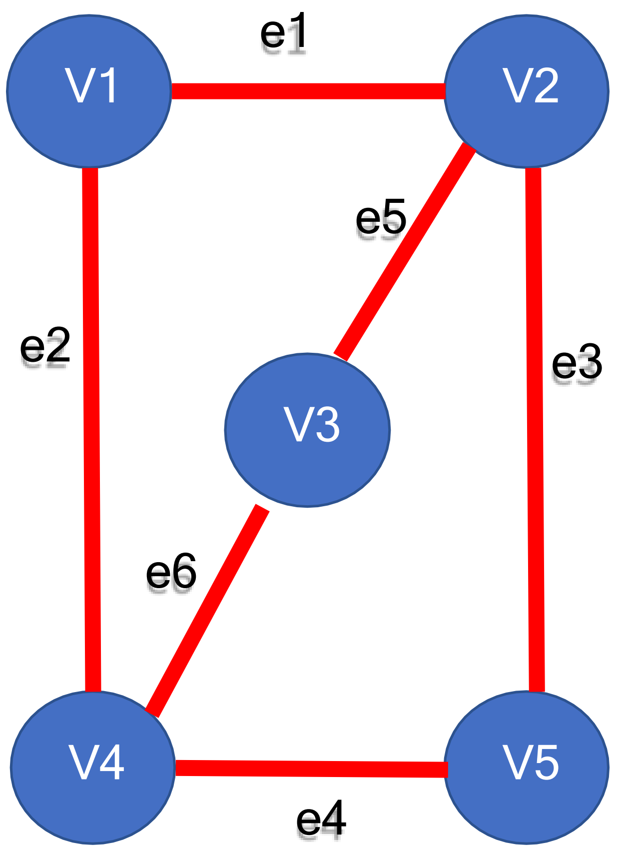
Undirected graph

- Chain vs Path
  - chain

A chain is a sequence of vertices connected by edges.

its length=number of edges used = number of vertices used minus one.

Example: {v1,v2,v3,v4,v5}
{v1,v2,v3}
{v1,v4,v5}

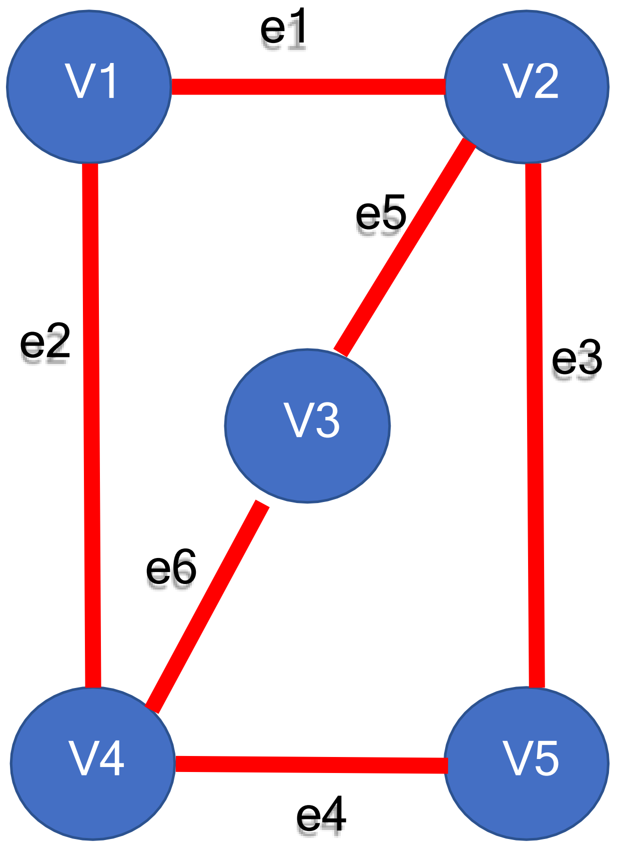


Undirected graph

- Elementary chain vs. simple chain

An elementary chain cannot visit the same vertex twice. A simple chain cannot visit the same edge twice.

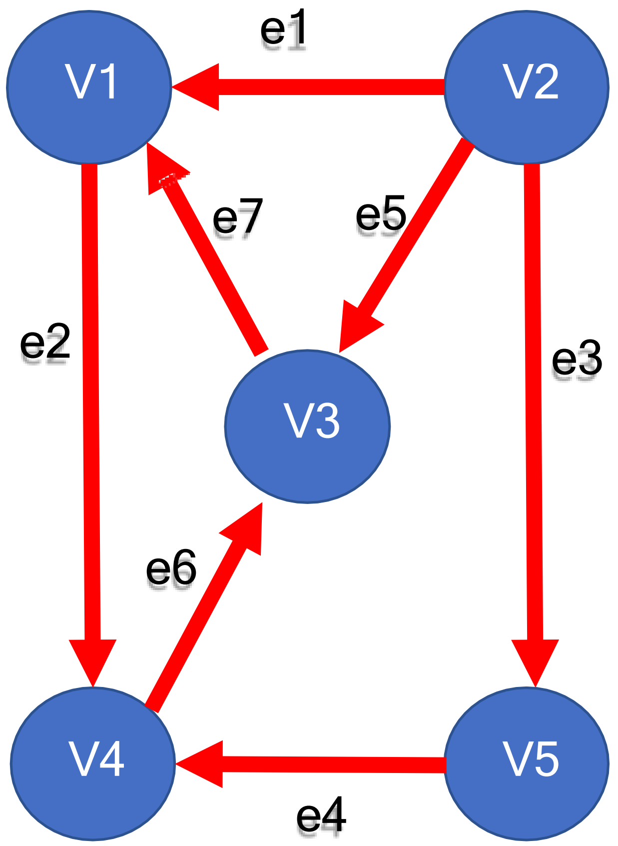
Example:
{v1,v2,v3,v4} is an elementary and a simple chain.



Directed graph

- Chain vs Path
  - path (*path*)

A path in a digraph is a sequence of vertices connected to each other by arcs. The path length is the number of arcs used, or the number of vertices minus one.



Directed graph

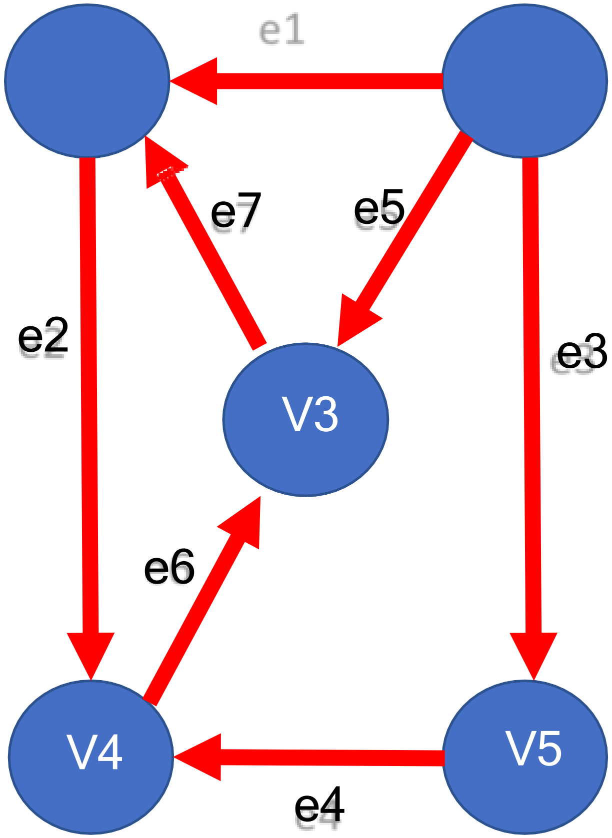
- Elementary path vs. simple path

simple path)

A simple path cannot visit the same arc more than once. An elementary path cannot visit the same vertex more than once.

{v2,v1,v4,v3,v1} is not an elementary path

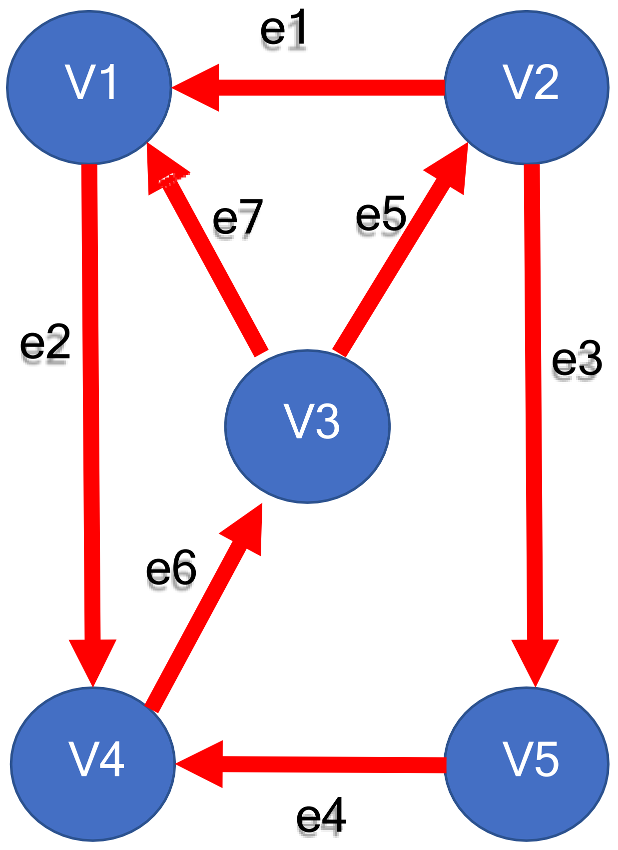
{e3,e4,e6,e7,e2} is a simple path but not elementary



Directed graph

- Circuit (*circuit*)

{v2,v1,v4,v3,v2}



A circuit is a simple closed path.

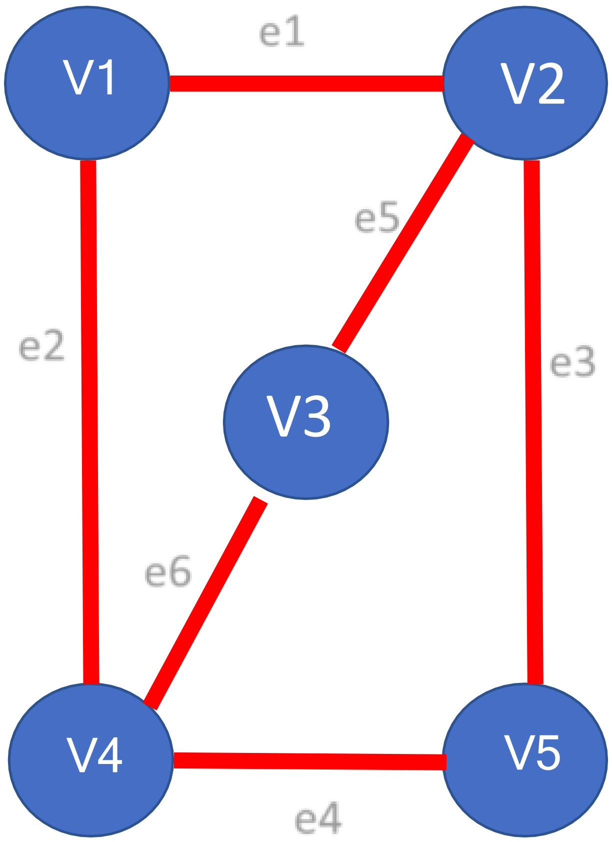
Undirected graph

- Cycle (*cycle*)

{v2,v1,v4,v3,v2} is a cycle= a simple closed chain

A cycle is a simple chain whose ends coincide.
We do not pass through the same vertex twice, except the one chosen as the starting and finishing vertices.

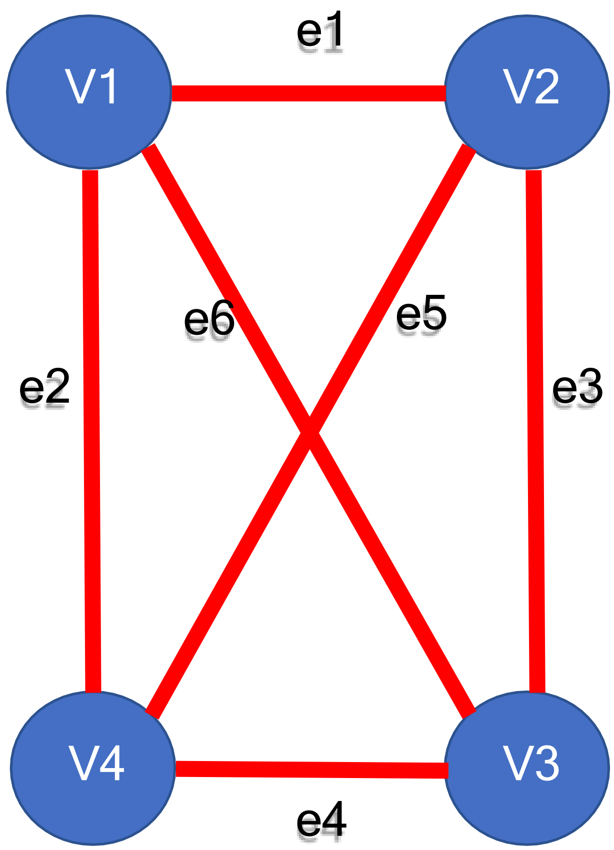
starting vertex = finishing vertex.



Complete graph

- *complete graph*

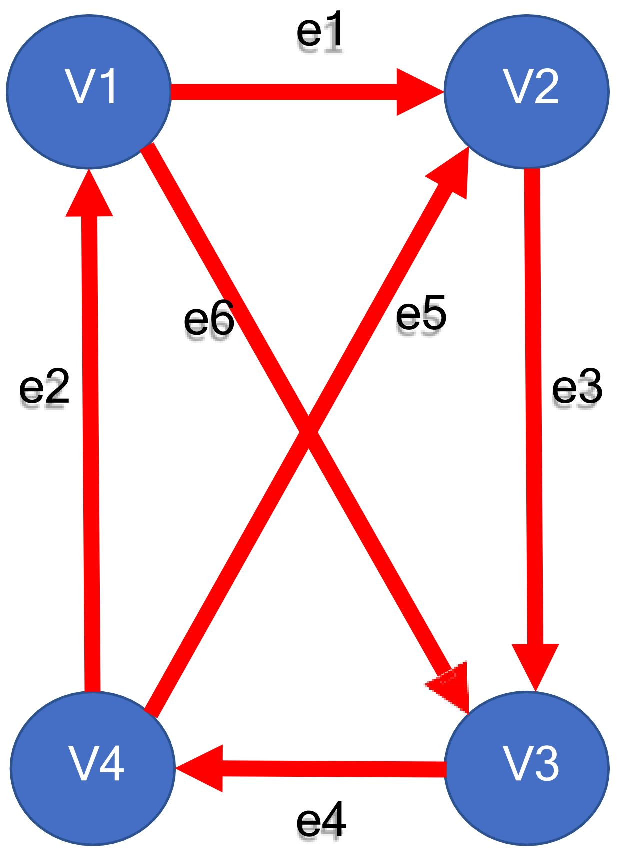
a finite graph G = (V,E) is said to be complete (*complete graph*) if all pairs of vertices are adjacent.



Tournament

- Tournament (**Complete Directed Graph**)

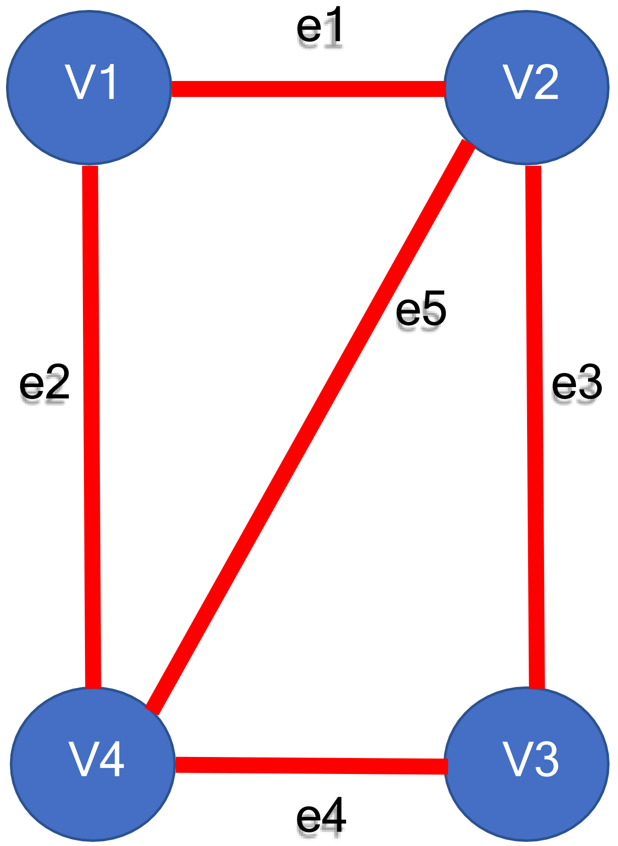
A graph G= (V,E) is called *Tournament* if G is oriented and complete.



Simple graph vs multigraph

- Simple graph (*simple graph*)

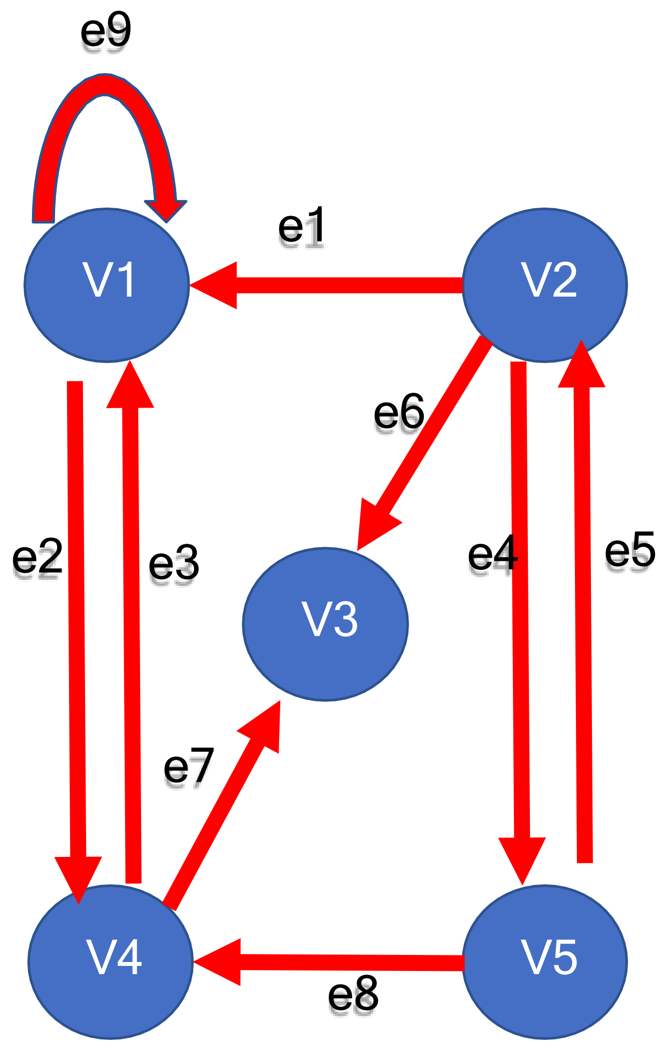
A finite graph G = (V,E) is said to be *simple*, if it does not contain a loop and if there is no more than one edge connecting two same vertices.



Simple graph vs multigraph

- Multigraph (*Multigraph*)

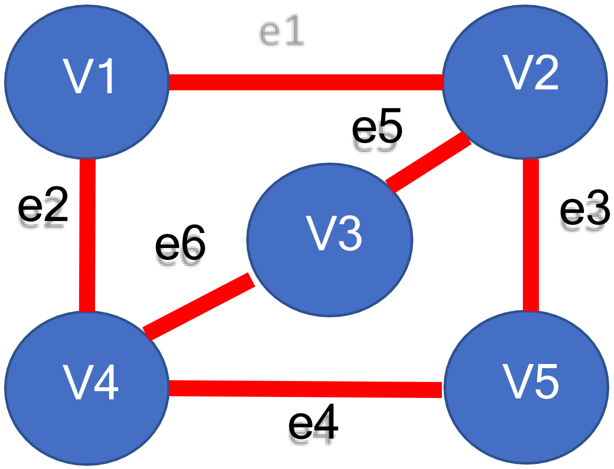
A finite graph G = (V,E) is called a multigraph if it contains loops and/or multiple edges connecting the same vertices.



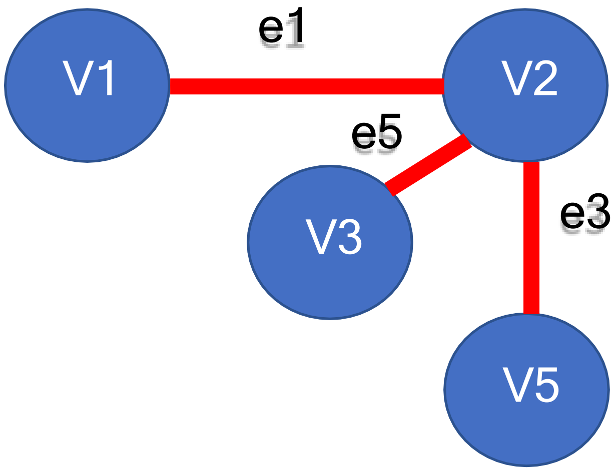
Subgraph

- Subgraph (*Induced subgraph*)

An *Induced subgraph* is obtained by removing from a graph vertices and all edges incident to those vertices.



Subgraph of G SubG =({V1,V2,V3,V5}, {e1,e5,e3})

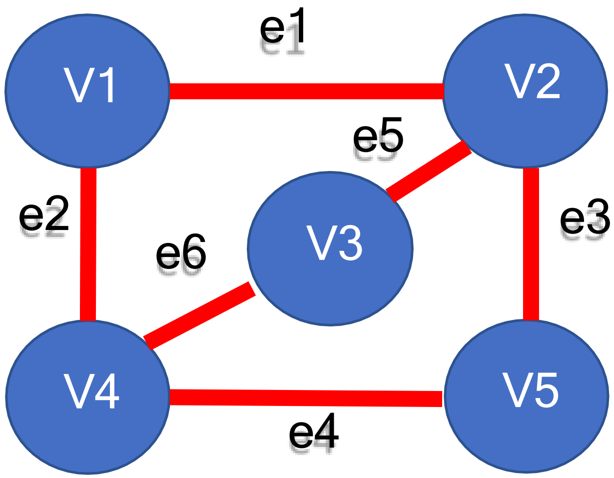


Partial graph

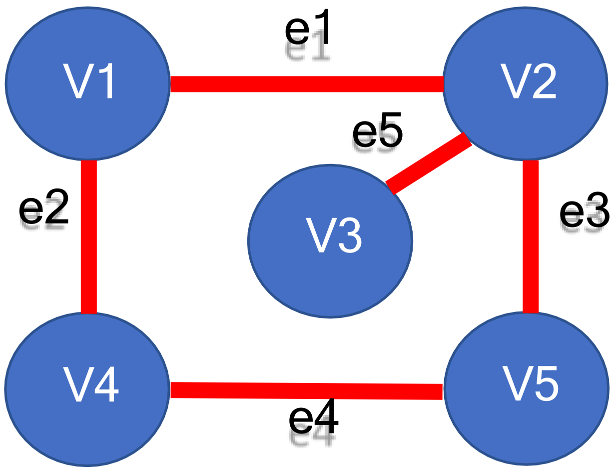
- Partial graph (*Spanning subgraph*)

A partial graph (*spanning subgraph*) is obtained by removing edges from a graph.

G=({V1,V2,V3,V4,V5},{e1,e2,e3,e4,e5,e6})



Partial graph of G partialG =({V1,V2,V3,V4,V5},
{e1,e2,e3,e4,e5})



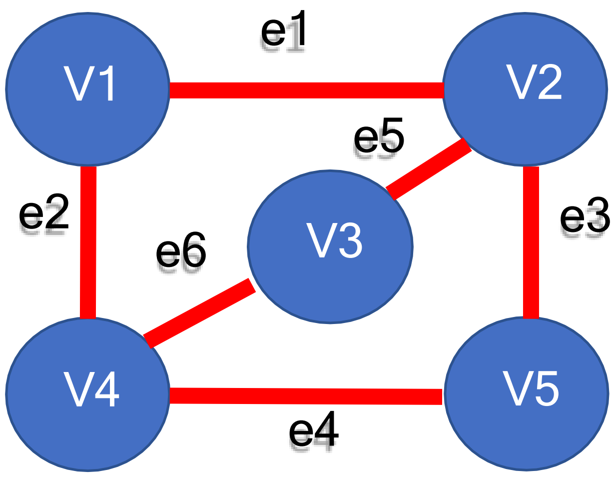
Eulerian

- *Eulerian graph/ path/chain/circuit/cycle*

A chain or cycle is said to be Eulerian if each edge of the graph appears exactly once. Paths and circuits of digraphs are said to be Eulerian under the same conditions.

A graph/digraph is said to be Eulerian if it admits an Eulerian cycle/circuit.

G=({V1,V2,V3,V4,V5},{e1,e2,e3,e4,e5,e6})



Eulerian chain of G = {e2,e1,e5,e6,e4,e3}

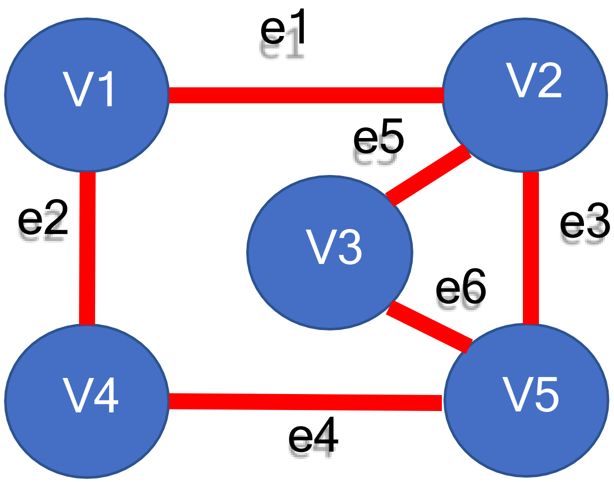
Hamiltonian

- *Hamiltonian graph/path/chain/circuit/cycle)*

A chain or cycle is said to be Hamiltonian if each vertex of the graph appears exactly once. Paths and circuits of digraphs are said to be Hamiltonian under the same conditions.

A graph/digraph is said to be Hamiltonian if it admits a Hamiltonian cycle/circuit.

G=({V1,V2,V3,V4,V5},{e1,e2,e3,e4,e5,e6})



G is Hamiltonian

because it admits a Hamiltonian cycle

{V1,V2,V3,V5,V4, V1}

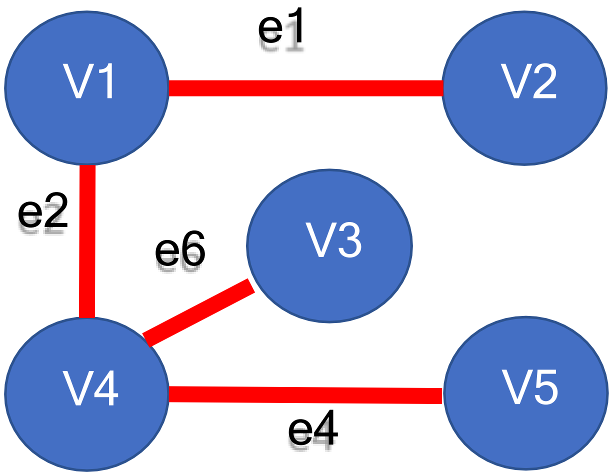
Hamiltonian cycle = All vertices are visited + each vertex is visited only once + starting vertex = ending vertex

Tree

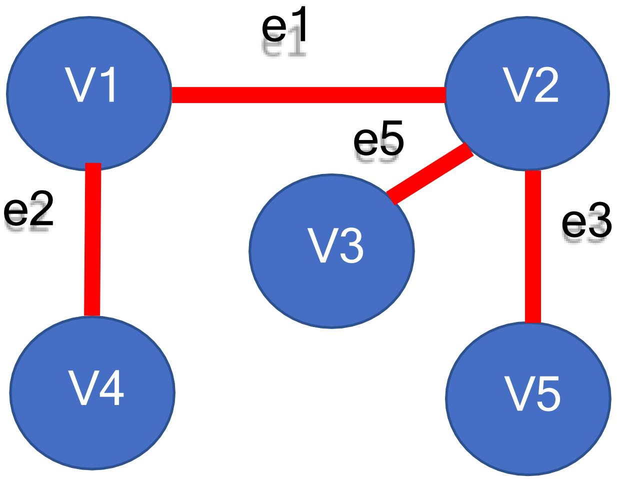
- *Tree*

A *Tree* is a connected graph containing no cycles.

G1=({V1,V2,V3,V4,V5},{e1,e2,e4,e6}) is a tree



G2 =({V1,V2,V3,V4,V5}, {e1,e2,e5,e3}) is a tree

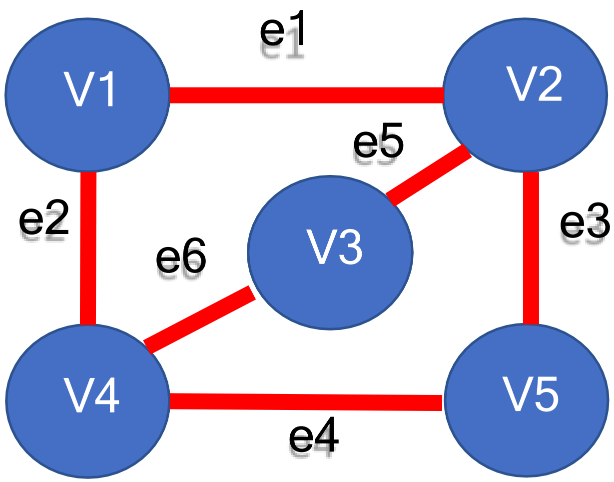


Covering tree (Spanning tree)

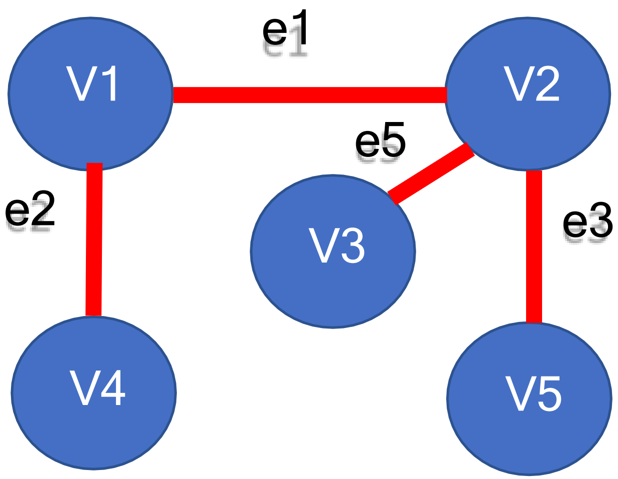
- Covering tree (*Spanning Tree*)

A covering tree (*Spanning Tree*) is a maximal subgraph of a graph containing no cycles (which is also a tree).

G=({V1,V2,V3,V4,V5},{e1,e2,e3,e4,e5,e6})



Spanning tree of G treeG =({V1,V2,V3,V4,V5},{e1,e2,e5,e3})



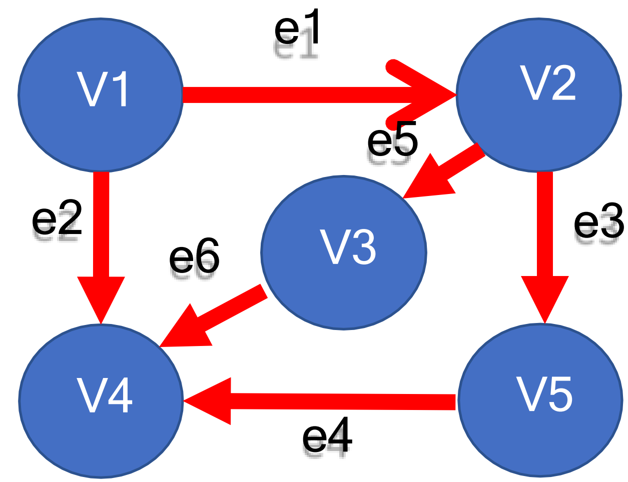
Rooted tree

- *Rooted tree*

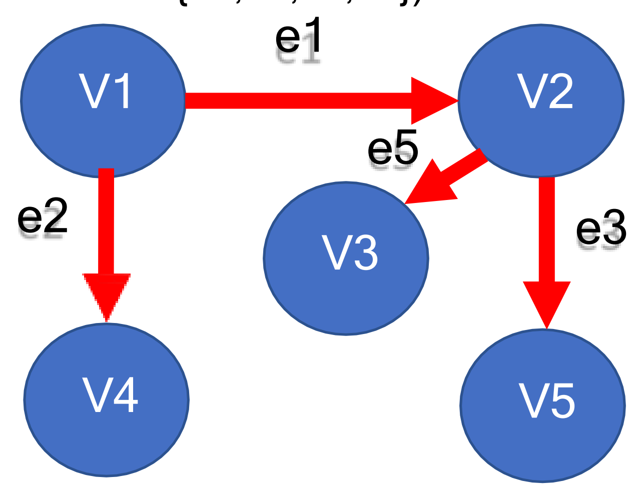
A *Rooted tree* is a tree with a distinguished or favored top r (*Root*).

example: RootedG is a tree with vertex V1 as root, because RootedG is a tree + there is one and only one path from V1 to all vertices.

G=({V1,V2,V3,V4,V5},{e1,e2,e3,e4,e5,e6})



RootedG =({V1,V2,V3,V4,V5}, {e1,e2,e5,e3})

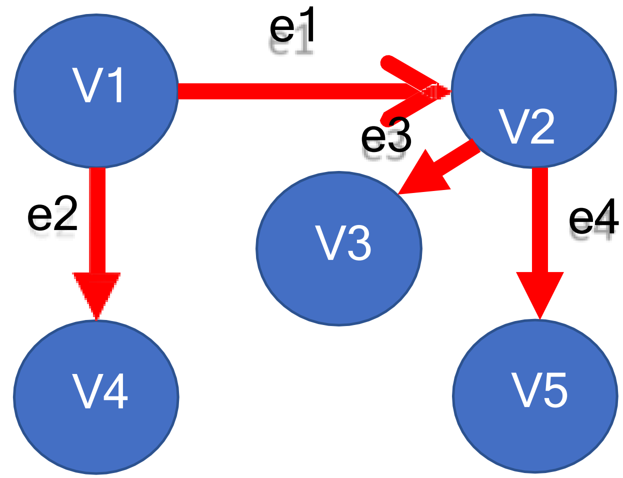


Forest

- *Forest*

A *Forest* is a set of trees.
A forest is a graph that does not contain cycles.
The related components of the forest are trees.
example: Tree 1 and Tree 2 are two components of a forest.

Tree1=({V1,V2,V3,V4,V5},{e1,e2,e3,e4})



Tree2 =({V1,V2,V3,V4},{e1,e2,e3})